Europäisches Patentamt	European Patent Office	Office européen des brevets

10.816.791
07.13.04

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.	The attached documents are exact copies of the European patent application described on the following page, as originally filed.	Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

03425211.4

CERTIFIED COPY OF
PRIORITY DOCUMENT

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 03425211.4
Demande no:

Anmeldetag:
Date of filing: 04.04.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

STMicroelectronics S.r.l.
Via C. Olivetti, 2
20041 Agrate Brianza (Milano)
ITALIE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

A method of implementing one-to-one binary function and relative hardware device, especially for a Rijndael S-box

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/ Classification internationale des brevets:

H03M13/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL PT RO SE SI SK TR LI

FIELD OF THE INVENTION

The present invention relates to a simple method and a relative fast hardware device for implementing an one-to-one binary function, which is particularly suited for realizing S-box devices performing the ByteSub operation of the Rijndael AES encryption/decryption algorithm.

BACKGROUND OF THE INVENTION

In devices implementing encryption/decryption algorithms it is necessary to perform binary functions over an input set of bytes to be encrypted/decrypted, for generating corresponding output bytes. Substantially these operations consist in an one-to-one binary function that is implemented by a logic circuitry, which is required to be fast, low power and small silicon area consuming.

Because of the importance of the Rijndael AES encryption/decryption algorithm, the problem to be solved is presented referring to this algorithm, but the same considerations hold *mutatis mutandis* also for any binary one-to-one function.

Brief overview of AES

In January 2, 1997, the National Institute of Standards and Technology (NIST) announced the beginning of the development of the Advanced Encryption Standard (AES). The overall goal was to develop a Federal Information Processing Standard (FIPS) that specified an encryption algorithm capable of protecting sensitive (unclassified) government information into the twenty-first century.

The formal call for algorithms was made on September 12, 1997. The algorithms were required to implement symmetric key cryptography as a block cipher and to support a block size of 128 bits and key sizes of 128, 192, and 256 bits. In August 20, 1998, NIST announced fifteen AES candidate algorithms at the First AES Candidate Conference, and solicited public comments on the candidates. A Second AES Candidate Conference was held in March 1999 to discuss the results

of the analysis that was conducted by the international cryptographic community on the candidate algorithms. In August 1999, NIST announced its selection of five finalist algorithms from the fifteen candidates. The selected algorithms were: MARS, RC6, Rijndael, Serpent and Twofish.

A lot of attention had been put on the complexity of the algorithms: a good AES algorithm was required to be easy to be implemented on general purpose processors and on reconfigurable hardware and light from a computational point of view.

NIST judged the Rijndael algorithm to be the best algorithm for the AES at the end of a very long and complex evaluation process in which all public comments, papers, verbal comments at conferences, and NIST studies and reports had been analyzed. The official announcement was made on October 2, 2000 and the standard was completed with the publication of FIPS-197 [1] on November 26, 2001.

Many ways of realizing devices for implementing efficiently the Rijndael AES encryption/decryption algorithm have been investigated. Papers (see for instance [2, 3]) that describe implementations of the AES algorithm on FPGAs (Field Programmable Gate-Arrays) are present in the technical literature. Few of them (see [4]) describe an implementation of the Rijndael algorithm on ASIC (Application-Specific Integrated Circuit) platforms.

A custom but still flexible implementation however would be desirable in high speed or embedded dedicated cores, in which fast and/or low-power computation is desirable.

The consumed silicon area for realizing an electronic circuit that performs the steps of the Rijndael algorithm is an important parameter for custom implementations. The smaller the consumed area, the lower the unit cost, and thus the higher the number of hardware devices produced on the same silicon die.

It has been observed [5] that the realization of the so-called S-box, which is a

hardware device that performs the byte substitution operation (ByteSub) contemplated by the algorithm, is critical for reducing the area consumption.

This operation is the bottle neck of the algorithm because it must be repeated many times and it is implemented by a one-to-one nonlinear binary function. Moreover, given that this function is nonlinear, it is not possible to realize a hardware device for performing it by using standard synthesis techniques.

The ByteSub operation is a composition of two binary functions defined on bytes. The first function is an inversion in the finite field (or Galois Field) $GF(2^8)$, which is a field composed of bytes, while the second is an affine function. Optimum security properties are obtained for the whole cipher system by combining in cascade these two functions (see for instance [6] and [7]).

The first function is more complex than the second from a computational point of view, because it behaves almost like a purely random function. For this reason, the S-box is generally synthesized starting from the complete truth table of the function. In practice, a behavioral table-like description is provided to a VHDL compiler, and the corresponding combinatorial function is extracted and synthesized with logic gates.

An alternative approach consists in considering the ByteSub operation as a function defined on the composite finite field $GF((2^4)^2)$. Substantially, the input byte is decomposed into two nibbles and the problem of inverting the first function is reduced to the problem of inverting a function in the inner field $GF(2^4)$ (which is smaller). This method has been originally proposed by Rijmen in [8], and further developed in [9]. See also [10], that contains performance estimations of the obtained circuit when implementing by the HCCMOS7 technology library of STMicroelectronics.

A further study on the optimization of the S-box has been proposed in [11], mainly addressing speed issues.

Another important issue in custom implementations is power consumption.

Although many techniques are described in technical literature for reducing power consumption at transistor level and at higher levels, still a hand-made, *ad hoc* analysis and design is often useful to produce low power custom units. Synthesis tools have features for power optimization, but still most of the power can be saved by human analysis and by giving the VHDL compiler a "good starting point".

Moreover, power optimization is often in contradiction with other design goals, such as small chip area and high speed, high frequency operation.

A low power approach for designing the Rijndael S-box is discussed in [12], which is a quite recent paper. The authors start from the compact implementation discussed in [8], exploit a PPRM technique (Positive Polarity Reed-Muller) and add delay chains in order to reduce power consumption and glitches of the architecture. This is the only paper dealing with a low-power S-box design.

OBJECT AND SUMMARY OF THE INVENTION

A method for implementing one-to-one binary functions defined on the Galois field $GF(2^8)$ is presented. This method is very useful for realizing fast and low power hardware devices whichever the binary function is. In particular, the method of the invention is particularly suited for realizing electronic circuits that implement the Rijndael encryption/decryption algorithm.

More precisely, object of the present invention is a method of generating output bytes corresponding to respective input bytes according to an one-to-one binary function, comprising the steps of

- decoding an input byte generating at least a bit string that contains only one active bit;
- logically combining the bits of the bit string according to the binary function for generating a 256-bit string representing a corresponding output byte;
- encoding the 256-bit string in a byte, obtaining the output byte.

This method may be implemented by a fast and small area consuming hardware device for generating output bytes corresponding to respective input bytes according to an one-to-one binary function, comprising

- a decoder of the input byte, generating at least a bit string that contains only one active bit;
- an array of logic gates input with the bit string, generating a 256-bit string by logically combining the bits of the input string according to the one-to-one binary function;
- an encoder input with the 256-bit string, generating the output byte.

The invention is more precisely defined in the annexed claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The different aspect and advantages of the invention will appear even more evident through a detailed description referring to the attached drawings wherein

Figure 1 depicts an embodiment of the hardware device of the invention;

Figure 2 illustrates a simple architecture of the decoder of Figure 1;

Figure 3 illustrates the preferred embodiment of the hardware device of the invention;

Figure 4 shows a detailed view of the left and right decoders of Figure 3;

Figure 5 compares power-delay performances of different AES S-box architectures with that of the S-box of the invention;

Figure 6 compares power-delay performances for different composite fields AES S-box architectures;

Figure 7 compares silicon area requirements of different AES S-box architectures;

Figure 8 compares the power consumption of the eight S-boxes implementing the DES algorithm, realized according to the method of the invention and with a table description technique.

DESCRIPTION OF AN EMBODIMENT OF THE INVENTION

In the ensuing description, the method of the invention is described referring to the AES S-box as an example. However, what is stated hereinafter does not depend exclusively on the particular binary functions of the Rijndael algorithm. By contrast, the method can be immediately extended to any one-to-one binary function. The new method admits also variants that achieve different levels of efficiency. They are all listed in order, starting from the simplest to the most sophisticated one.

As it has been said hereinbefore, a careful analysis of a boolean function that must be implemented often leads to high power savings. The analysis can be made automatically by the design tools, or by the designer himself at a high level of complexity.

The S-box implements a nonlinear byte substitution function that has good statistical properties: the input-output correlation is kept as low as possible and the function is very similar to a random function.

Recently, an analysis that sheds some shadows on the designers' claim for robustness of the nonlinear part has been published (see [13]). Moreover, there is no evidence that this result can be useful for area or power reduction in hardware designs, although it can be a good starting point for a cryptanalytic attack of the cipher algorithm it is contained in.

It has been noticed that the ByteSub operation performed by the S-box is nothing but a permutation of all the elements of the Galois finite field $GF(2^8)$, i.e. it is a byte permutation. Hence, this operation is an one-to-one mapping.

Therefore, instead of computing the input-output function in a standard way, according to the invention it is possible to do it by decoding the input byte generating at least a bit string containing only 1 active bit, for example a 256-bit string or two 16-bit strings, then rearranging the bits according to the one-to-one binary function obtaining a 256-bit string, as depicted in Figure 1, which may be

done with a null power dissipation. Finally, the output byte is obtained by encoding back to a byte the 256-bit string obtained after having changed the order of the bits.

Note that the rearrangement of the wires is not strictly necessary, because the same result may be obtained by using an array of logic gates generating the bits of the 256-bit input it in the encoder.

The decoder may be simply designed using AND and NOT logic gates, which are present in the technology library. Different options are available: a naïve solution would consists in designing it as a single binary tree using only two-inputs AND cells, as in Figure 2. Note that each input bit wire must be available in normal and complemented form and it enables a single subtree. The number of AND logic gates that are switching when the input sequence changes is at most 14, because the worst case occurs when the two decoded lines belong to two different sub-trees; in this case 7 gates switch off and 7 gates switch on. The maximum number of NOT gates that may switch is 8.

The drawback of this implementation consists in that power consumption is highly affected by high net fan-outs and the input bit wires, must face an increasing fan-out as they are closer to the output of the decoder. For instance: bit 0 and 1 have a fan-out of only 4 AND gates, while bit 7 has a fan-out of 256 AND gates (one half of them pertaining to the normal form and the other half to the complemented form).

Another solution consists in providing a single level of 256 eight-inputs AND components, one per each decoded line. This solution is faster (latency is only that of one eight-input AND gate, which is normally faster than 6 cascaded two-inputs AND gates), but each input line has a fan-out of 256 gates. In this case just 2 AND gates are switching when the input sequence changes.

The preferred solution consists in realizing a hardware device for implementing a one-to-one binary function with a two-level architecture, as shown in Figure 3.

The Left Decoder block and the Right Decoder block have the same internal structure. Their task is decoding the most (left) and the least significant (right) nibbles of the input byte, respectively, and generating respective left and right 16-bit strings each having only one active byte by decoding the input nibble.

An array of 256 two inputs AND gates is provided and is input with bits of the 16-bit strings; each AND gate is connected to an unique combination of the left and right decoder output lines.

For instance, consider the internal line representing the Galois field element 0x63 (i.e. the byte 0x63); this line is connected to the output of the AND gate, which in turn is connected to the 7^{th} output line from the left decoder and to the 4^{th} output line of the right decoder. This internal line is active only when the left decoder receives the value '6' and switches active its 7^{th} output line, and when the right decoder receives the value '3' and switches active its 4^{th} output line.

Given that the inputs of S-box may be considered random distributed byte values, it holds:

- each of the left and right decoder input lines has a switching probability of 1/2 over a long working period;
- each of the 16 left and right decoder output lines has a switching probability of 1/16 over a long working period;
- each of the 256 internal lines has a switching probability of 1/256 over a long working period.

The internal structure of the left and right decoders is represented in Figure 4: this is a normal architecture for a priority-less decoder.

An immediate advantage of the proposed architecture is that the maximum fan-out of nets is fixed to 16 gates: keeping low the fan-out of the nets leads to a low power consumption. Another advantage is that the latency of the decoder is equal to the latency of only one NOT gate and of two AND gates.

The encoder component is more area consuming and brings more switching activity to the global circuit.

Preferably, it is constituted only of OR gates. In practice, every encoder output line is a logic OR between exactly 128 internal lines. For each output bit, the OR-tree is practically implemented by 3 levels of components: the first level is made of 16 eight-inputs OR gates, the second level is made of 2 eight-inputs OR gates and the third level, which provides the output bit, is just a two-input OR gate.

Such an architecture is commonly used for encoders: although it is quite area consuming, since the encoder has 256 input lines, it can be very fast because the latency is equal to that of 2 eight-input OR gates and 1 two-input OR gate. Moreover, the maximum number of switching gates consequent to a switching input line is fixed at 16 eight-inputs OR gates and at 8 two-inputs OR gates; this corresponds to the worst case when all the 8 output lines are switching at the same time.

The maximum fan-out of the encoder is 8, and specifically it is the fan-out of the input line (1 out of 256) in correspondence of active bits of the output byte.

The hardware device of the invention, may be used for realizing a S-box device for the Rijndael algorithm making the implemented one-to-one binary function be the function representing the ByteSub operation.

Performances of a "Decode-Encode" (D&E) AES S-box architecture realized according to the scheme of Figure 3 starting from VHDL files, will be discussed. In this way, the validity of the proposed method for the efficient synthesis of one-to-one combinatorial networks is demonstrated.

The synthesis of the AES S-box has been carried out using the design tool Synopsys Design Compiler version 2001.08, and the HCCMOS7 (by STMicroelectronics) technology library, featuring a 0.25 μm process working at 1.8 V core voltage.

Figure 5 depicts power-delay diagrams from:

- Purely behavioral description (S-box bhv), where the S-box truth table is simply given to the design tool. This description is useful to test the compiler's capability of optimizing the circuit with built-in techniques. Note that the VHDL code would be the same in case a ROM was present; however for relatively small structures like the S-box, the ROM table is often allocated as a combinational logic and optimized by the silicon compiler.

- Mixed structure (S-box D&E mixed), wherein the Decode-Encode structure is explicitly given to the compiler, but the decoder and the encoder blocks are described in a behavioral mode. This description may be useful to test the capability of the compiler of using logic blocks inside the specific technology library to map these circuits.

- Structural Decode-Encode description (S-box D&E struct), wherein the architecture described above is explicitly given to the compiler (VHDL is written with a structural approach). The simplest components, such as AND and OR gates, are described in behavioral code and form the basis for the structural description.

It is worth noticing that the proposed Decode-Encode S-box architecture (S-box D&E struct) behaves very well in the region between 1.6ns and 3ns time latency; power consumption in this region is about 50% smaller than the power consumption of the S-box obtained with a purely behavioral description. Both architectures can reach a speed limit of 1.53ns using this specific technology library, but the proposed architecture is not worthy at full speed. It is interesting to note that the compiler is not able to map automatically fast power-efficient encoders and decoders, as can be seen from the D&E mixed line. The performances of this architecture are better than those obtained with the purely behavioral approach, and this can be seen as a partial benefit of the Decode-Encode structure.

The composite fields architecture has been coded as well, and the power delay

figures are reported in Figure 6, together with a single mapping from the CHES2002 low-power implementation, which is basically a modification of the composite fields architecture. It is worth noticing that for the latter case it was not possible to model delay chains with the HCCMOS library, which are an essential part of the architecture. Results are expected to be better than the standard composite field case when delay chains are correctly modeled (true for power consumption, but delays will be very similar). As may be noticed from the graph, power-delay figures are worse for the composite fields architecture, which by contrast is a convenient architecture as far the area occupation is concerned.

Figure 7 compares minimum area occupation figures for the different S-box architectures.

In order to prove the general validity of the method of the invention for implementing any one-to-one binary function, other cases are examined.

Let us notice that if the values of the function implemented by the AES S-box have to be changed for security reasons, it is sufficient to change the arrangement of the internal lines of the S-box of the bit strings obtained by decoding the input byte without increasing the power consumption.

Another possibility consists in inserting multiplexers in the inner switch, in order to support the direct and inverse ByteSub simply with an additional control line. This will lead to important area savings, given that encryption and decryption operations are not used at the same time.

It is possible to optimize the S-boxes of other different cryptographic algorithms as well. The "Decode-Encode" scheme is still applicable for any hardware device implementing an one-to-one binary function.

For instance, the method of the invention may be applied to the DES (Data Encryption Standard) algorithm [14], as well as to the Kasumi algorithm [15], which is the candidate encryption algorithm for the new UMTS infrastructure.

Figure 8 shows the power consumption figures of the eight S-boxes for

performing the DES algorithm: these components have been synthesized first starting from a table description[1], and then by means of the proposed method. The power saving is evident, and a further unexpected but favorable result consists in that the eight S-boxes dissipate substantially the same power because they differ only in the way the wires are permuted. This is an important feature, because it allows to eliminate "hot spots" in the design and allows a more regular power consumption of the whole circuit.

[1] Note that this is the only way to describe the DES S-boxes, since they are not derived from arithmetic operations.

REFERENCES

[1] *"Announcing the ADVANCED ENCRYPTION STANDARD (AES)"* – Federal Information Processing Standard Publication 197, 2001

[2] M. McLoone and J.V. McCanny, *"High performance single-chip FPGA Rijndael algorithm"*, Proceedings of CHES 2001

[3] V. Fischer and M. Drutarovsky, *"Two methods of Rijndael implementation in reconfigurable hardware"*, Proceedings of CHES 2001

[4] H. Kuo and I. Verbauwhede, *"Architectural optimization for a 1.82Gbits/sec VLSI implementation of the AES Rijndael algorithm"*, Proceedings of CHES 2001

[5] P. R. Shaumont, H. Kuo, I. Verbauwhede, *"Unlocking the Design Secrets of a 2.29Gb/s Rijndael Processor"*, Proceedings of DAC 2002

[6] K. Nyberg, *"Differentially uniform mappings for cryptography"*, Proceedings of EUROCRYPT 1993

[7] J. Daemen, V. Rijmen, *"AES Proposal: Rijndael"*, 1999

[8] V. Rijmen, *"Efficient Implementation of the Rijndael S-BOX"*

[9] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, P. Rohatgi, *"Efficient Rijndael Encryption Implementation with Composite Field Arithmetic"*, Proceedings of CHES 2001

[10] M. Macchetti, G. Bertoni, *"Hardware Implementation of the Rijndael Sbox: a Case Study"*, to appear in the ST Journal of System Design

[11] S. Morioka and A. Satoh, *"A 10 Gbps Full-AES Crypto Design with a Twisted-BDD S-BOX Architecture"*, Proceedings of ICCD 2002

[12] S. Morioka and A. Satoh, *"An Optimized S-box circuit architecture for low power AES design"*, Proceedings of CHES 2002

[13] J. Fuller and W. Millan, *"On Linear Redundancy in the AES SBOX"*, 2002

[14] "Data Encryption Standard", FIPS PUB 46-1, 1988

[15] Website: www.3gpp.org

CLAIMS

1. A method of generating output bytes corresponding to respective input bytes according to an one-to-one binary function, comprising the steps of

decoding an input byte generating at least a bit string that contains only one active bit;

logically combining the bits of said bit string according to said binary function for generating a 256-bit string representing a corresponding output byte;

encoding said 256-bit string in a byte, obtaining said output byte.

2. The method of claim 1, wherein said second 256-bit string is obtained by carrying out the steps of

subdividing the input byte in a left nibble (L) and a right nibble (R);

decoding the left nibble (L) and right nibble (R) in a left 16-bit string and a right 16-bit string, respectively, each containing only one active bit;

logically combining the bits of said 16-bit strings according to said binary function for generating said 256-bit string.

3. The method of claim 1, wherein

said input byte is decoded in a corresponding auxiliary 256-bit string;

said first 256-bit string is obtained by changing the order of the bits of said auxiliary 256-bit string according to said binary function.

4. The method of claim 1, wherein said one-to-one binary function represents the ByteSub operation of the Rijndael AES encryption/decryption algorithm.

5. The method of claim 2, wherein each bit of said second 256-bit string is obtained by ANDing among them bits of said 16-bit strings.

6. A hardware device for generating output bytes corresponding to respective input bytes according to an one-to-one binary function, comprising

a decoder of the input byte, generating at least a bit string that contains only one active bit;

an array of logic gates input with said bit string, generating a 256-bit string by

logically combining the bits of the input string according to said one-to-one binary function;

an encoder input with said 256-bit string, generating said output byte.

7. The hardware device of claim 6, wherein

said decoder is composed of a left decoder and a right decoder input with a left nibble (L) and a right nibble (R) of the input byte, and generating a left 16-bit string and a right 16-bit string, respectively, each containing only one active bit;

said array of logic gates generates said 256-bit string as logic combination of bits of said 16-bit strings.

8. The hardware device of claim 7, wherein said array of logic gates is an array of 256 AND gates, each generating a respective bit of said 256-bit string by ANDing bits of said 16-bit strings.

9. The hardware device of claim 7, further comprising

an array of multiplexers each input with bits of said 16-bit strings and driven by selection signals and generating a respective intermediate bit fed to said array of logic gates;

said logic gates generating bits of said 256-bit string by logically combining said intermediate bits.

10. The hardware device of claim 6, wherein

said decoder of the input byte generates a corresponding auxiliary 256-bit string;

said array of logic gates generates said first 256-bit string by changing the order of the bits of said auxiliary 256-bit string in input according to said one-to-one binary function.

11. An electronic circuit performing the Rijndael AES encryption/ decryption algorithm, comprising a hardware device (S-box) according to claim 6 wherein said one-to-one function corresponds to the ByteSub operation of said algorithm.

"A METHOD OF IMPLEMENTING ONE-TO-ONE BINARY FUNCTION AND RELATIVE HARDWARE DEVICE, ESPECIALLY FOR A RIJNDAEL S-BOX"

ABSTRACT

A method for implementing one-to-one binary functions defined on the Galois field GF(2^8) is presented. This method is very useful for realizing fast and low power hardware devices whichever the binary function is, and comprises the steps of

- decoding an input byte generating at least a bit string that contains only one active bit;
- logically combining the bits of the bit string according to the binary function for generating a 256-bit string representing a corresponding output byte;
- encoding the 256-bit string in a byte, obtaining the output byte.

This method may be implemented by a fast and small area consuming hardware device for generating output bytes corresponding to respective input bytes according to an one-to-one binary function, comprising

- a decoder of the input byte, generating at least a bit string that contains only one active bit;
- an array of logic gates input with the bit string, generating a 256-bit string by logically combining the bits of the input string according to the one-to-one binary function;
- an encoder input with the second 256-bit string, generating the output byte.